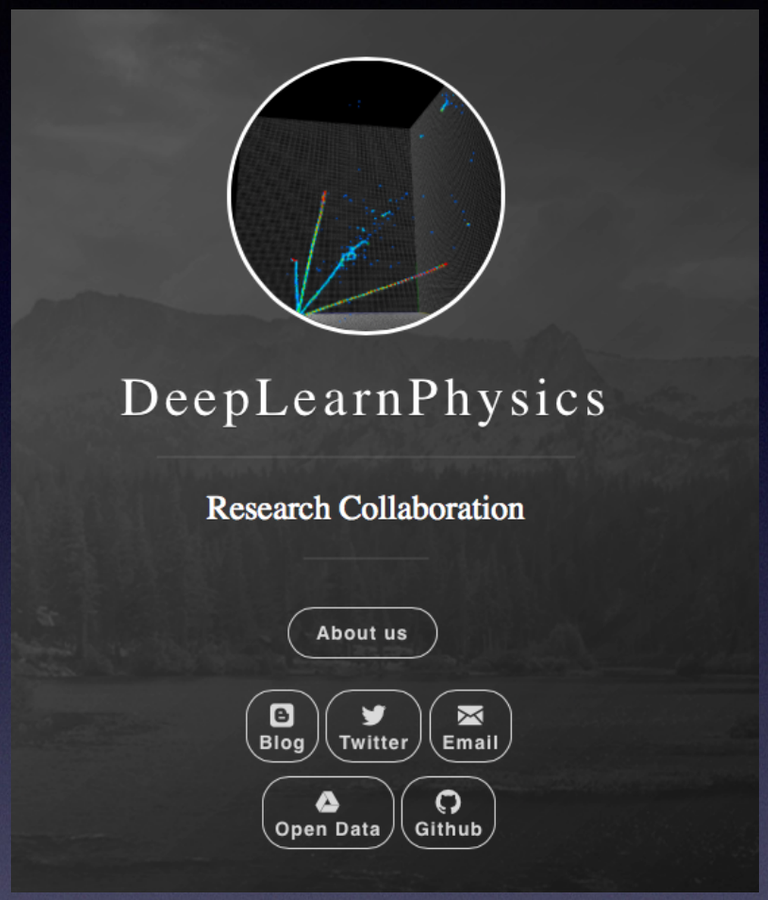
DeepLearnPhysics



- Group of physicists mainly from neutrino TPC experiments
- Share software tools + open data
- Meetings/Blog posts to share experience, discuss problems, etc.
- Interface with conferences, workshops, other similar groups

Interested in? You're welcome to join!

Deep Learning Application: Practice

1. Getting Started: MNIST + Tensorflow
2. Singularity: mobility of software stacks
3. About U-ResNet
4. Training U-ResNet
5. Overview: what it takes to apply DL in research

MNIST + Tensorflow

Open up Google Colaboratory

We run this notebook.

Singularity

- We use singularity container today
- Singularity is not installed on the system?
  - You can pull & build in 4 lines
  - No system install needed to run singularity
  - Use your account! (no sudo needed)

```
VERSION=2.4.2
wget https://github.com/singularityware/singularity/releases/download/$VERSION/singularity-$VERSION.tar.gz
tar xvf singularity-$VERSION.tar.gz
cd singularity-$VERSION
mkdir install
./configure —prefix=$PWD/install
make install
./install/bin/singularity
```

- Running a container as simple as:

```
$ singularity exec shub://DeepLearnPhysics/larcv2-singularity:ubuntu16.04-larcv_develop bash
```

  - Example: checkout this asciinema

... Shameful But I Need You To Do ...

- Follow these steps

Deep Learning Application Practice

What many people get stuck on:

- "I have done MNIST examples but I don't know how I can interface my data. It's in C++ software format."
- "I just pickled my data (or CSV-ed, you brave person) but my training is so slow."

My Point

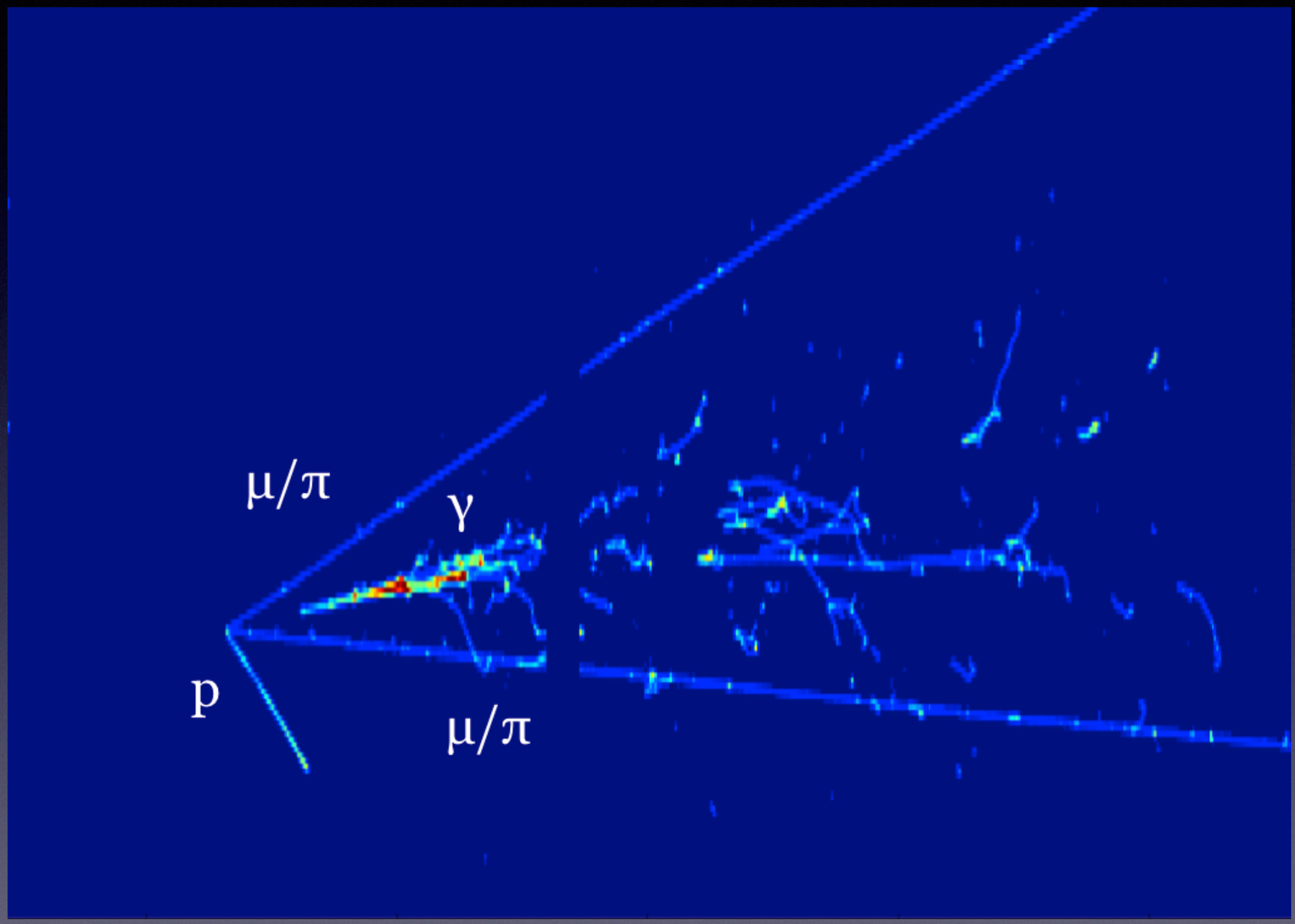
It is rare to hear people getting stuck on MNIST example. Typically practical problems rise in "trying to do the same with my own data."

Deep Learning Application Practice

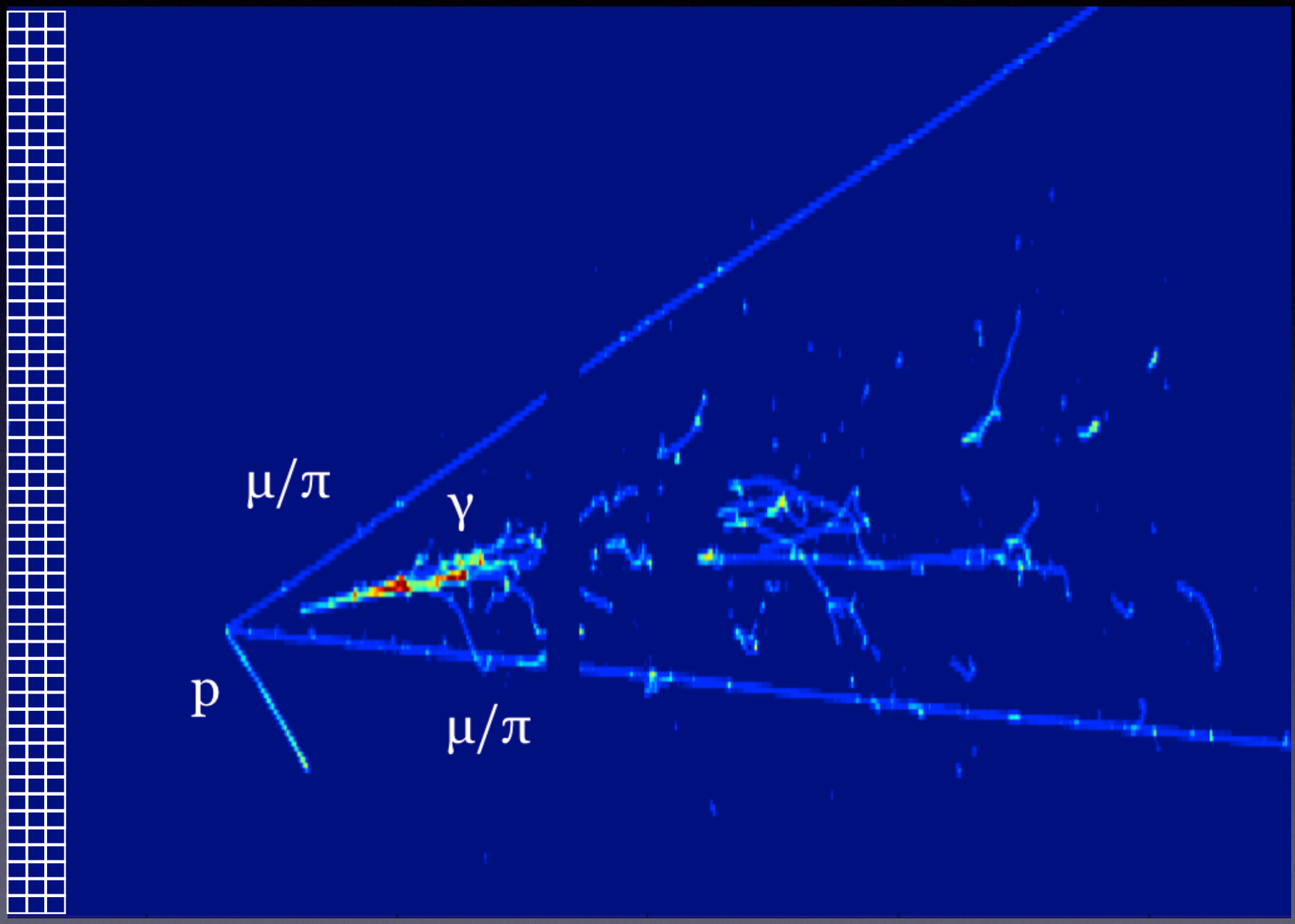
Three points

- **Data/File format** suited for your DL algorithms
  - image data format for CNNs
  - sparse data format for LArTPCs
- **Algorithms** to fill such data format
  - Input: your native (experiment software) data
  - Output: suited data format (+ file if you want to save it)
- **Interface to DL software**
  - May be written in Python/C++/CUDA
  - Different options, more on later slides

Data Format for LArTPC



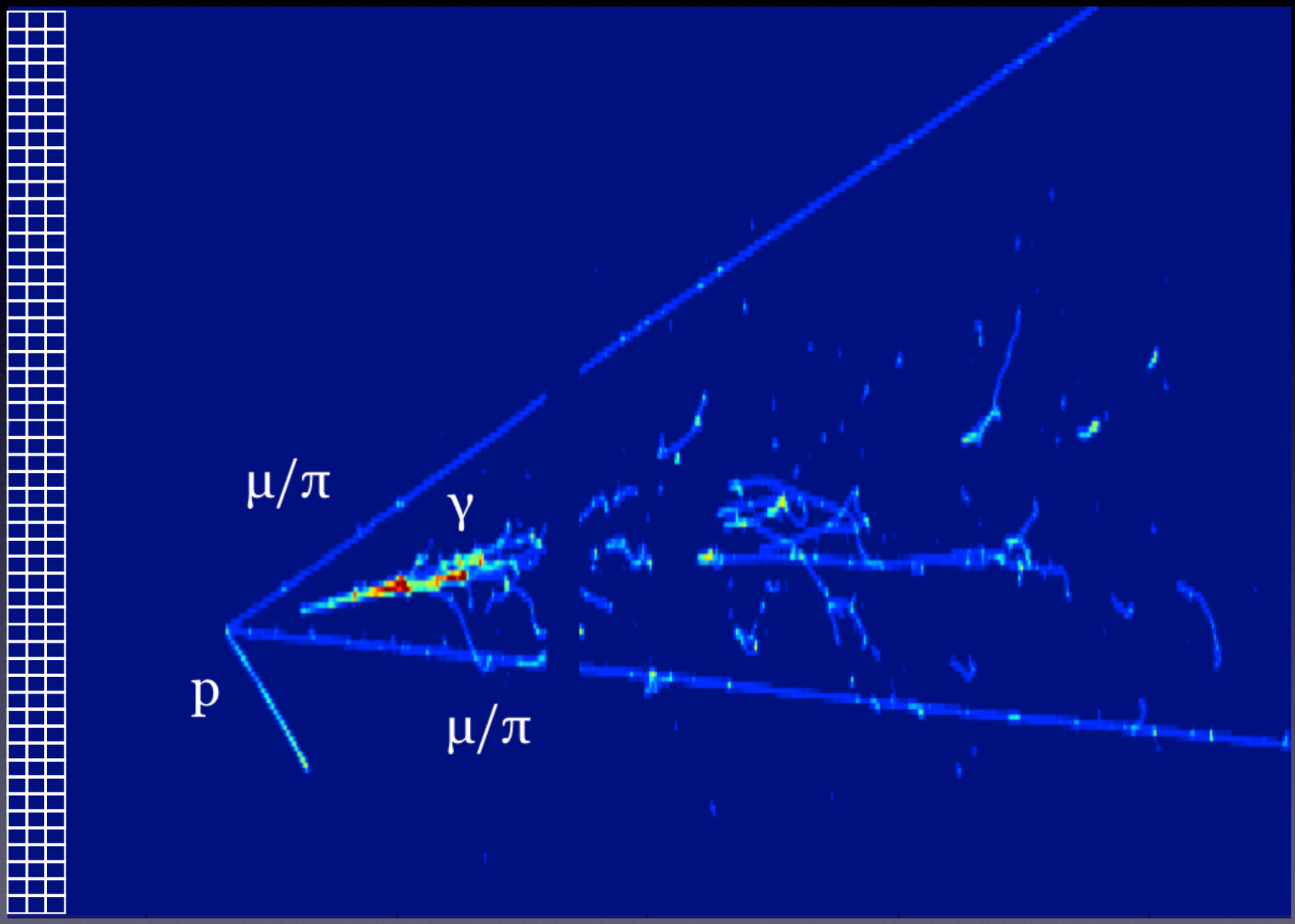
Data Format for LArTPC



std::vector<float>
std::vector<float>
std::vector<float>

... 8256 of std::vector<float> each with length 9600
(~80 Mpx photograph filled with mostly = 99% zeros)

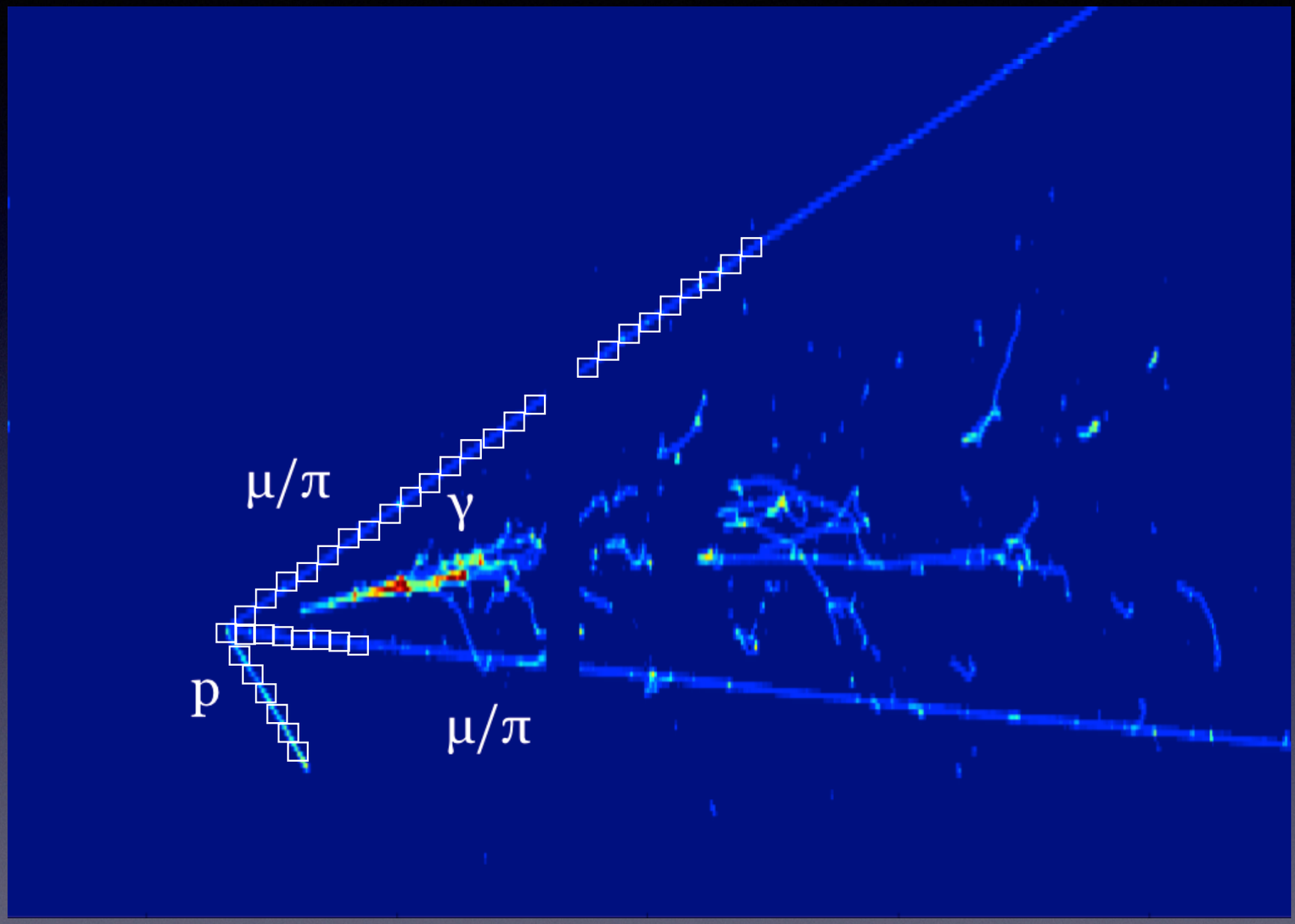
Data Format for LArTPC



std::vector<float>

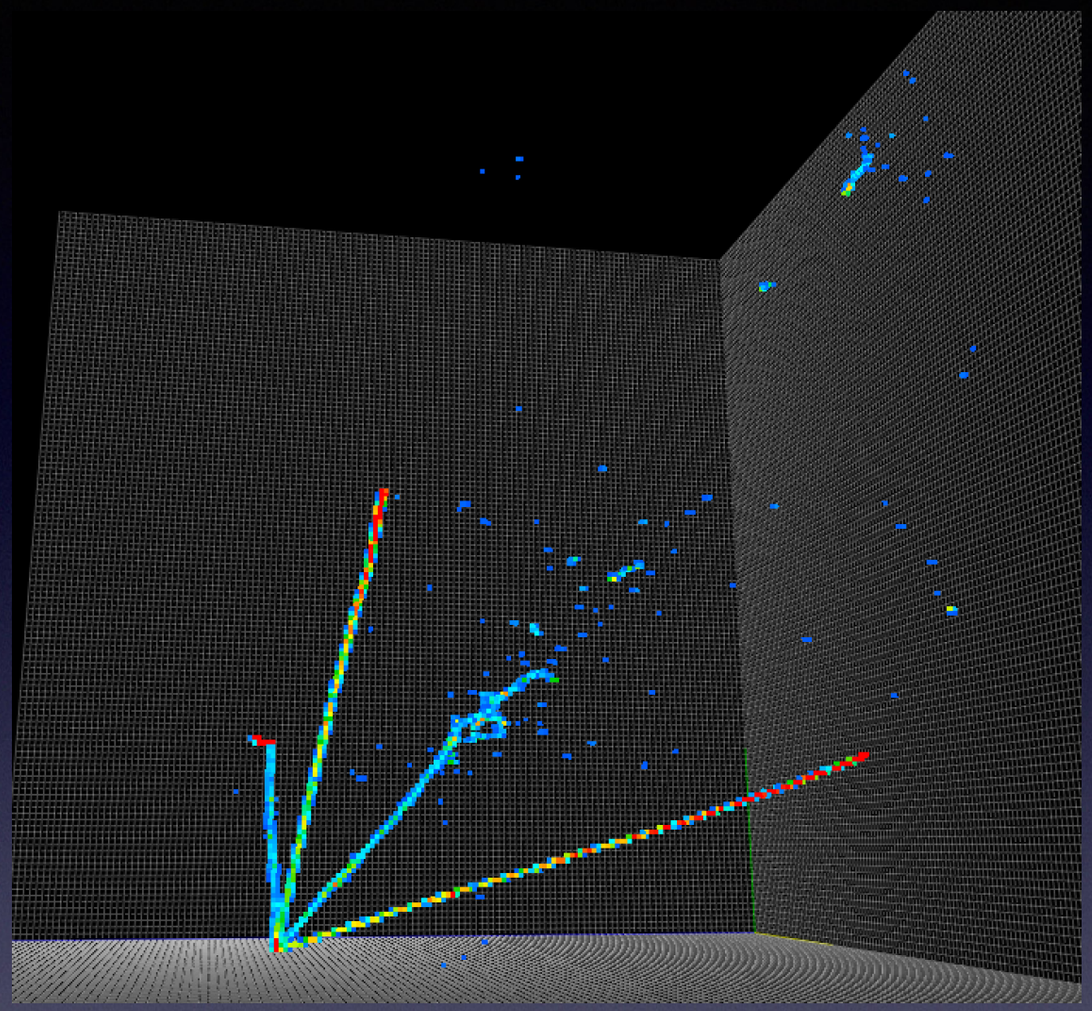
Option 1: single, contiguous array
Note: still lots of zeros though compressed well. Easily convertible to other formats (numpy array, cv::Mat, etc.)

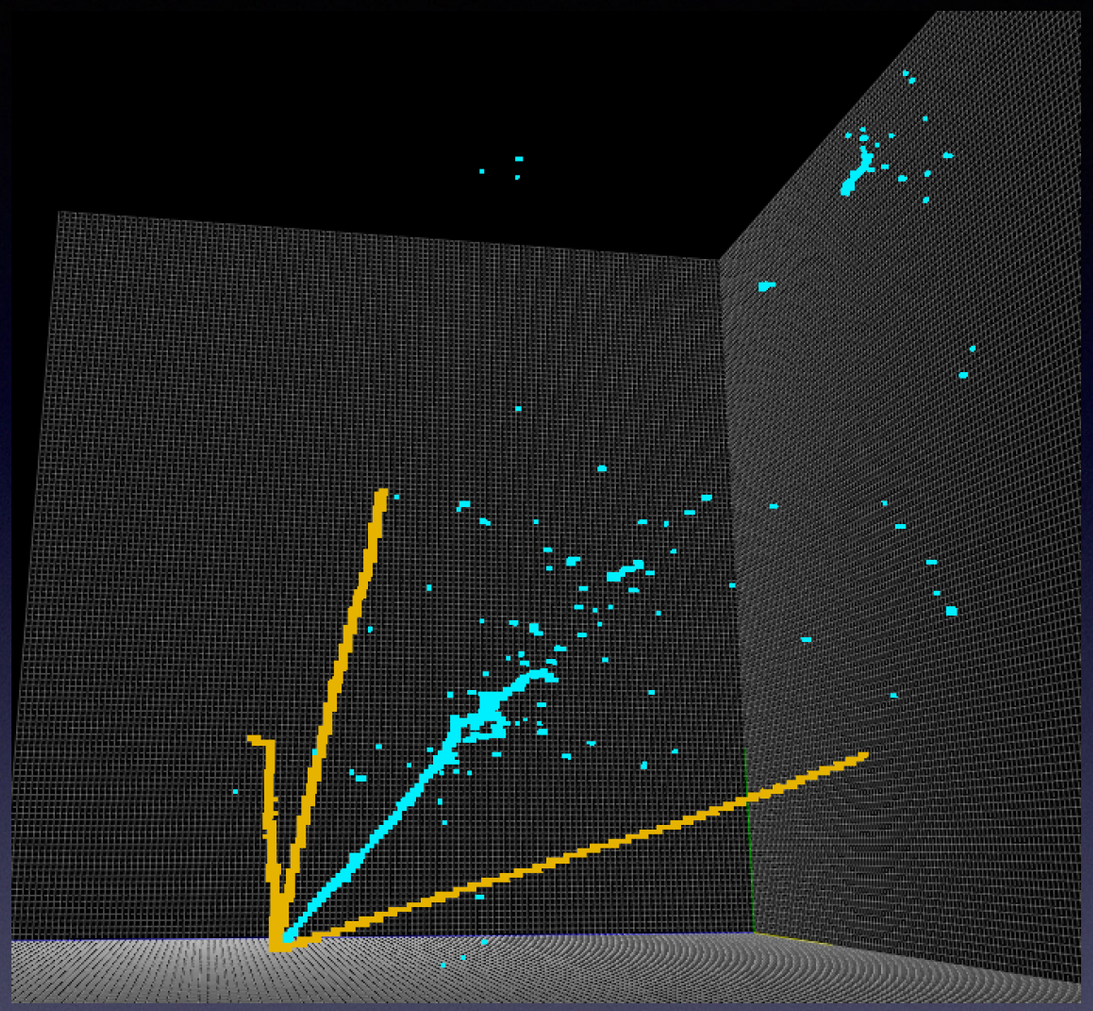
Data Format for LArTPC



Option 2: sparse vector (zero-suppressed)
Note: need CPU to interpret sparse tensor as a contiguous array but saves disk space a lot.

Algorithms

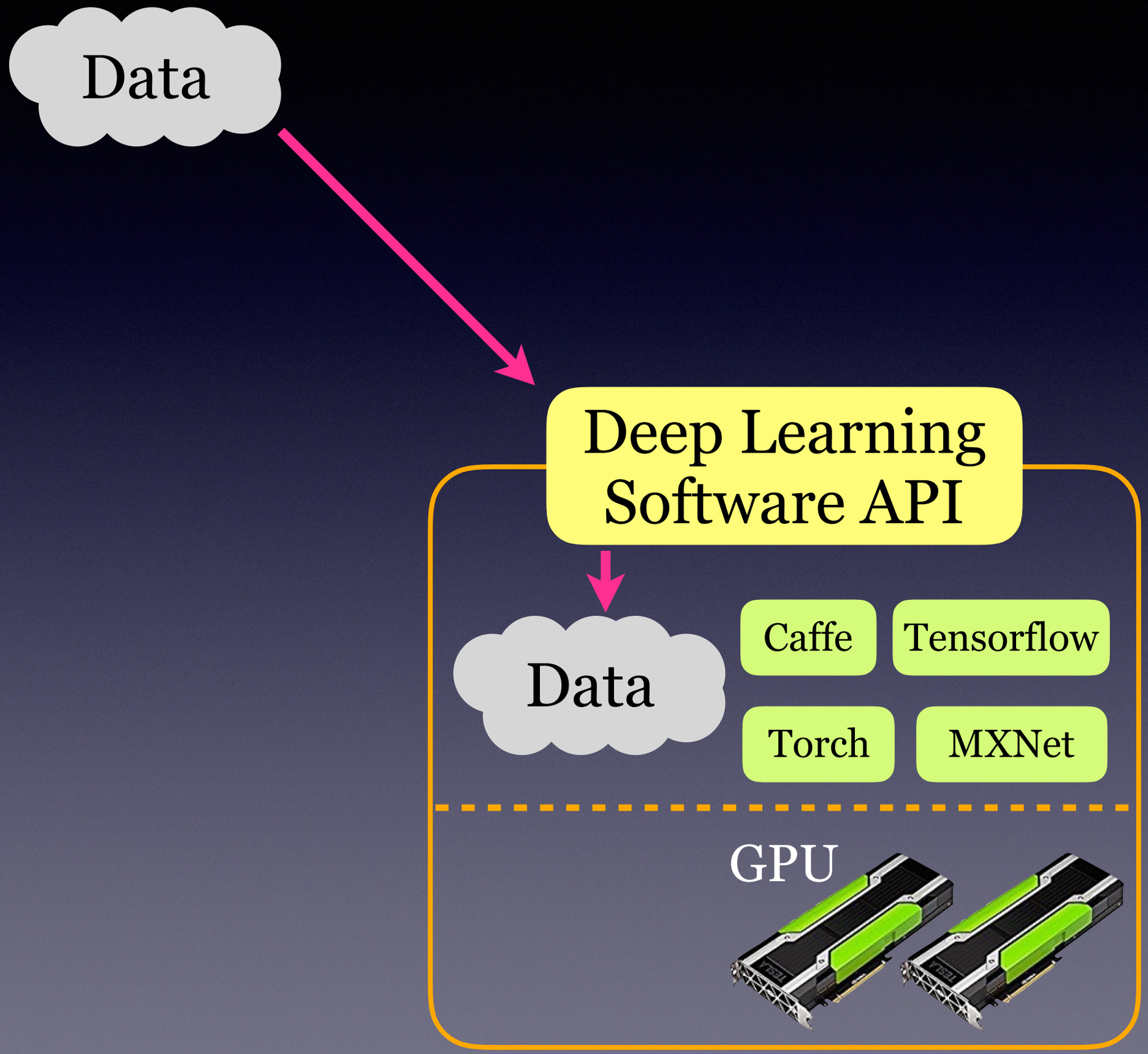




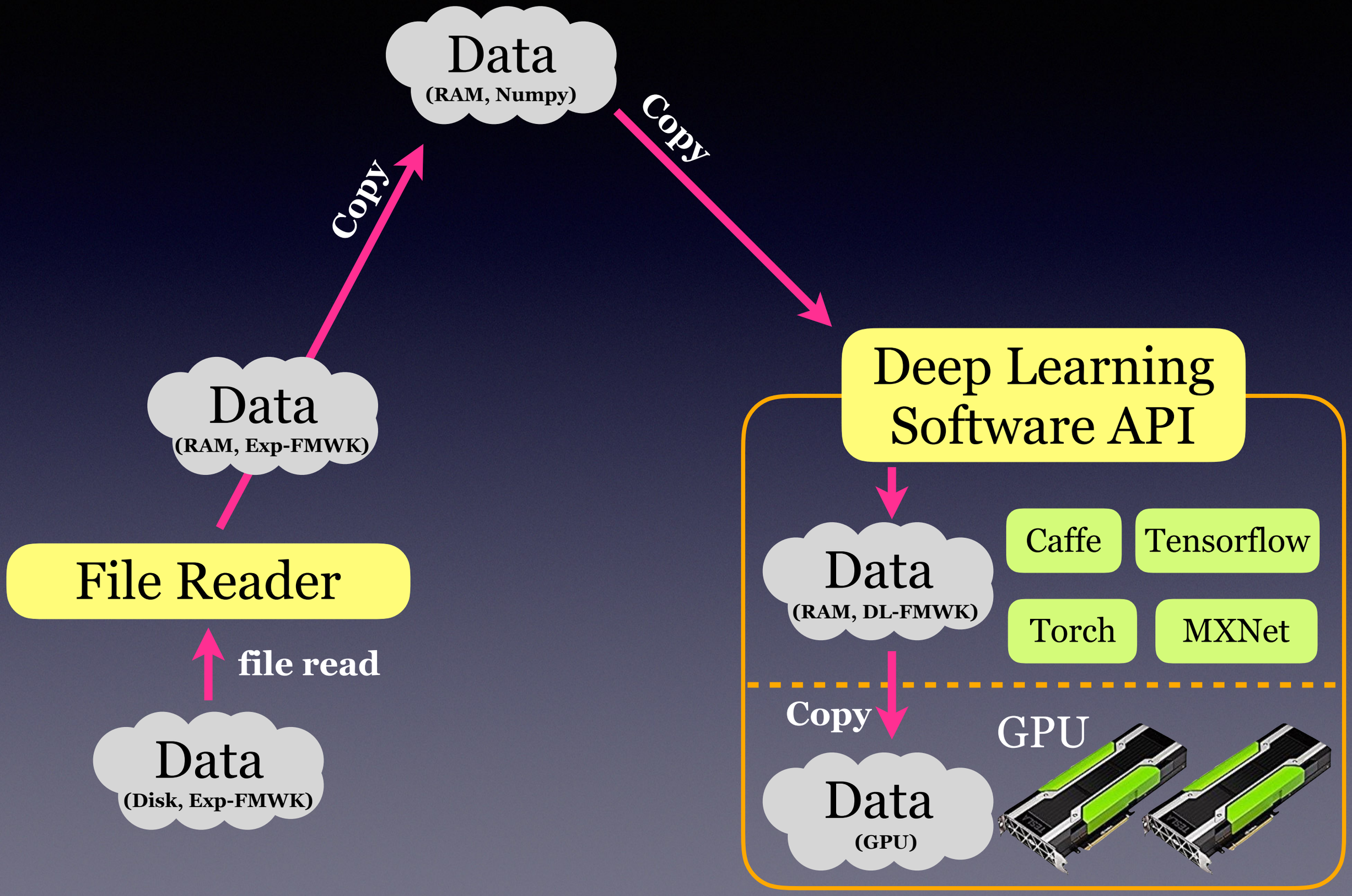
Data engineering is very important

- No matter what awesome network you make, it cannot learn beyond data you prepared
- Algorithm to extract important features and fill data
- Algorithm to “pre-process” data before training

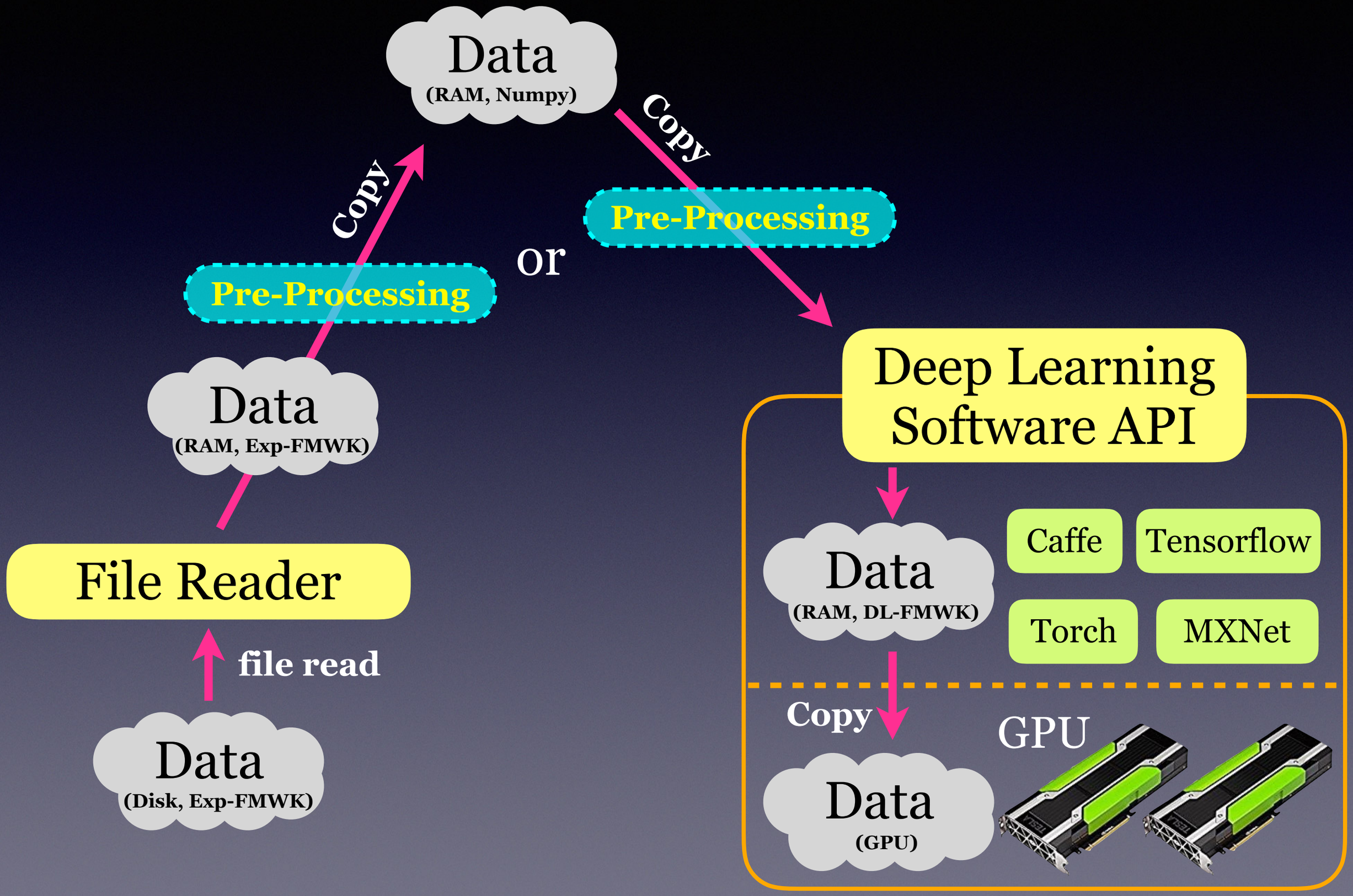
DL Software Interface



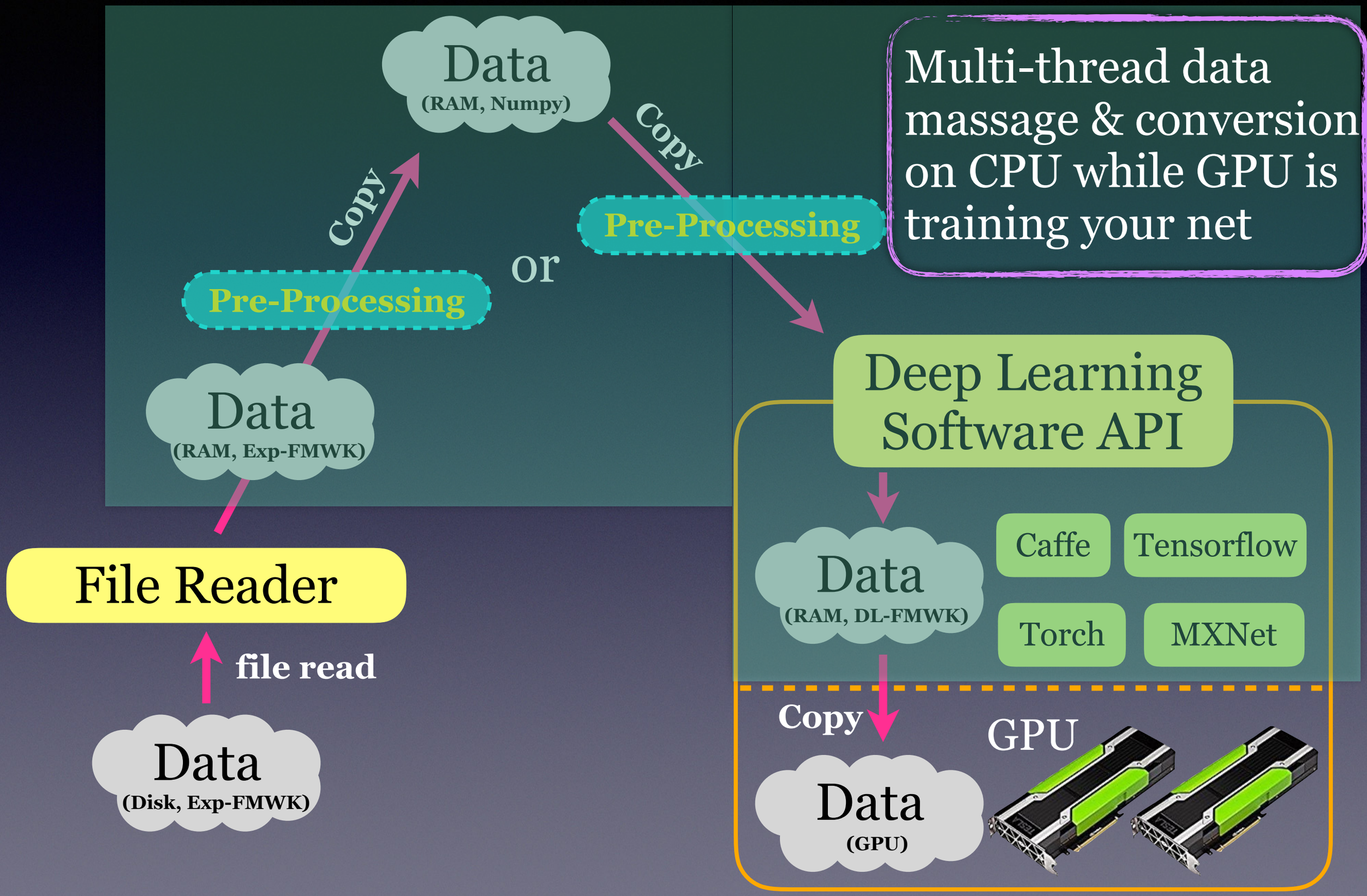
DL Software Interface



DL Software Interface



Optimization Example 1



Optimization Example 2

Skip Python layer and directly copy data into DL data structure

Threaded
Pre-Processing
Data
(RAM)

Threaded
Pre-Processing
Data
(RAM)

Threaded
Pre-Processing
Data
(RAM)

Converter
(threaded)

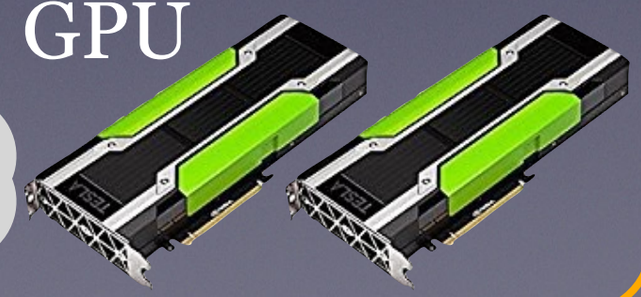
Deep Learning Software API

Copy
Data
(RAM, DL-FMWK)

Caffe
Tensorflow
Torch
MXNet

Copy
Data
(GPU)

GPU



File Reader

file read

Data
(Disk, Exp-FMWK)

Optimization Example 3

Write data file reader using DL software API (usually C++/CUDA)

Or

Use DL software API to write data file (native DL framework format)